Tampereen yliopisto

Aku Suvanto

# TILALLISTEN SOVELLUSTEN SKAALAAMINEN KUBERNETES-FEDERAATIOISSA

Kandidaatintyö
Informaatioteknologian ja viestinnän tiedekunta
Tarkastajat: Petri Kannisto
Elokuu 2022

# TIIVISTELMÄ

Aku Suvanto: Tilallisten sovellusten skaalaaminen Kubernetes-federaatioissa
Kandidaatintyö
Tampereen yliopisto
Tietotekniikka
Elokuu 2022

---

Ohjelmistokehityksessä suositaan nykyään entistä enemmän mikropalveluita, jotka toteutetaan konttiteknologian avulla. Mikropalveluiden tehokas ja automatisoitu hallinta on palvelun laadun ja tehokkuuden ylläpitämisen kannalta tärkeää. Automatisoitua hallintaa varten on luotu useita ohjelmistoja, joista eräs on Kubernetes. Kuberneteksen avulla yhden tai useamman mikropalvelun voi pakata yhdeksi hallittavaksi yksiköksi, kapseliksi, joka edustaa joko kokonaista palvelua tai jotakin sen osaa. Kuberneteksen ja sitä tukevan KubeFed-ohjelmiston avulla kapseleiksi pakattuja palveluita voidaan jakaa suoritettavaksi useille eri Kubernetes-klustereille yhdeltä pääklusterilta.

Tämän tutkielman tarkoitus on selvittää, kuinka tilallisten sovellusten skaalaaminen toteutetaan tällaisessa hajautetussa klusteriympäristössä. Tämä tutkielma on toteutettu kirjallisuuskatsauksena. Aineistoa on haettu useista alan tietokannoista, sekä lisäksi mainittujen ohjelmistojen dokumentaatiosta. Käytettyjen lähteiden valinnassa on painotettu uusimpia lähteitä mahdollisuuksien mukaan. Tutkielma jakaantuu kolmeen osa-alueeseen. Ensimmäiseksi tutkielmassa tarkastellaan Kuberneteksen ydinkonsepteja sekä tilallisuuden hallintaa. Seuraavaksi tutkielmassa tutustutaan klustereiden toimintaan ja tyyppeihin. Viimeisenä työssä tutkitaan, kuinka sovellusten skaalaaminen toteutetaan Kuberneteksessa.

Tutkielmassa havaittiin, että KubeFed-ohjelmiston avulla voidaan toteuttaa hajautettu federoitu klusteri, jossa tilallisten sovellusten skaalaaminen onnistuu pääosin samalla tavalla kuin yhden klusterin järjestelmissä. Federoidussa klusterissa pääklusteri vastaa kaikkien osaklustereiden konfiguraatioista. KubeFed-ohjelmiston avulla federoidulle klusterille voidaan määrittää federoituja versioita olemassa olevista Kuberneteksen rajapintatoiminnoista. Federoituja rajapintatoimintoja käyttämällä pääklusteri voi luoda ja hallita kapseleita kaikilla osaklustereilla. Rajapintatoimintojen avulla voidaan myös määrittää pääklusteri hallitsemaan sovellusten skaalaamista tai vaihtoehtoisesti määrittää tietty osaklusteri skaalaamaan niitä itsenäisesti.

Avainsanat: Kubernetes, KubeFed, skaalaaminen, klusterit, mikropalvelut

# SISÄLLYSLUETTELO

1. Johdanto . . . . . 1
2. Kubernetes . . . . . 2
2.1 Kapselit . . . . . 2
2.2 Jalkautukset . . . . . 3
2.3 Tilallisuuden hallinta . . . . . 3
2.3.1 StatefulSet ja taltiot . . . . . 4
2.3.2 Elinkaaren hallinta . . . . . 5
3. Klusterit . . . . . 6
3.1 Erilliset useamman klusterin järjestelmät . . . . . 6
3.2 Keskitetyt useamman klusterin järjestelmät . . . . . 7
3.3 Federoidut useamman klusterin järjestelmät . . . . . 8
4. Skaalaaminen . . . . . 9
4.1 Horizontal Pod Autoscaler . . . . . 9
4.2 Cluster Autoscaler . . . . . 10
5. Yhteenveto . . . . . 12
Lähteet . . . . . 13

# LYHENTEET JA MERKINNÄT

| | |
|---|---|
| CNCF | Linux Foundationin projekti konttiteknologian kehityksen edistämiseksi (engl. Cloud Native Computing Foundation) |
| EKS | Amazonin Kubernetes-alusta (engl. Amazon Elastic Kubernetes Service) |
| GKE | Googlen Kubernetes-alusta (engl. Google Kubernetes Engine) |
| HPA | Automaattisesta skaalaamisesta vastaava rajapintatoiminto (engl. Horizontal Pod Autoscaler) |
| RSP | Kapselin kopion sijoittamisesta vastaava rajapintatoiminto (engl. Replica Scheduling Preference) |
| YAML | Merkintäkieli (engl. YAML Ain't Markup Language) |

# 1. JOHDANTO

Ohjelmistokehityksen piirissä on jo useamman vuoden ajan ollut havaittavissa huomattavaa siirtymää poispäin monoliittisista sovelluksista kohti mahdollisimman pieniä mikropalveluita, jotka yhdessä muodostavat toimivan sovelluksen. Mikropalveluarkkitehtuuri helpottaa ohjelmiston ylläpitoa, sillä koko sovelluksen uudelleenpakkaamisen sijaan voidaan päivittää vain muutoksia kaipaavaa mikropalvelua. Mikropalveluiden tai muiden kontteihin pakattujen palveluiden tehokas käyttö vaatii kuitenkin hallintajärjestelmiä, joista eräs on Kubernetes.

Toistaiseksi monissa ympäristöissä on todettu riittäväksi pitää nämä hallintajärjestelmät rajoitettuina yhteen klusteriin, erityisesti jos palvelua pitää julkaista vain yhdellä maantieteellisellä alueella. Useampaa erillistä klusteria voidaan yleisesti myös hallita niin, etteivät ne tiedä toistensa olemassaolosta. Kubernetes kuitenkin tarjoaa työkaluja, joiden avulla useampia klustereita voidaan liittää yhdeksi kokonaisuudeksi, jolloin niiden automaattinen hallinta helpottuu.

Tämän työn tarkoituksena on vastata kysymykseen, miten tilallisia sovelluksia skaalataan Kubernetes federaatioissa. Työssä käydään läpi Kuberneteksen toimintaa, miten erilaiset klusterirakenteet eroavat toisistaan ja miten skaalaaminen federaatioympäristössä toteutetaan.

Työ on toteutettu kirjallisuuskatsauksena. Lähteiden haussa on hyödynnetty Andor- ja Google Scholar-hakutyökaluja sekä lisäksi IEEE Xplore tietokantaa. Työssä on myös käytetty Kuberneteksen dokumentaatiota. Hakusanoina on käytetty termejä "Kubernetes", "Kubefed", "Scaling", "Distributed Clusters", "Federated Clusters"ja "Stateful Applications". Tutkittava aihe on jatkuvasti kehittyvä, joten uusimmille lähteille on annettu tietoa etsiessä eniten painoarvoa. Käytetyt lähteet ovat englanninkielisiä ja niissä esiintyville termeille on pyritty etsimään sopivat suomenkieliset käännökset.

Luvussa 2 esitellään Kuberneteksen toimintaa, peruskäsitteitä, kapseleita, jalkautuksia sekä tilallisuuden hallintaa. Luvussa 3 eritellään erilaisia moniklusterirakenteita. Luvussa 4 esitellään Kuberneteksen ja KubeFedin tarjoamia työkaluja skaalaamisen toteuttamiseen. Viimeinen luku on yhteenveto työstä.

# 2. KUBERNETES

Kubernetes on avoimen lähdekoodin ohjelmisto, joka on suunniteltu kontteihin (engl. container) pakattujen ohjelmien hallintaan ja skaalaamiseen (”Kubernetes”, 2022). Kubernetes pohjautuu Googlen sisäiseen käyttöön kehitettyyn Borg-hallintajärjestelmään ja sisältää useita parannuksia edeltäjäänsä nähden (Verma et al., 2015). Nykyisin Kuberneteksen kehityksestä vastaa Cloud Native Computing Foundation (CNCF), jolle Google luovutti Kubernetes-projektin sen saavuttaessa 1.0-versionsa (”TechCrunch - As Kubernetes Hits 1.0, Google Donates Technology To Newly Formed Cloud Native Computing Foundation”, 2015).

## 2.1 Kapselit

Kapselit (engl. pod) ovat pienin Kuberneteksen jalkautettava (engl. deployable) yksikkö. Yksittäinen kapseli koostuu yhdestä tai usein useammasta palvelusta, jotka on sijoitettu kontteihin (engl. container). (”Kubernetes Documentation - Pods”, 2021). Kontti on virtuaalikoneen tapainen virtualisoitu ympäristö, joka sisältää kaikki palvelun tarvitsemat riippuvuudet, kuten esimerkiksi ohjelmointikielen tulkin tai ohjelman tarvitsemat kirjastot. Virtuaalikoneista poiketen kontit eivät kuitenkaan sisällytä mukaansa koko käyttöjärjestelmää, vaan hyödyntävät isännän (engl. host) käyttöjärjestelmää ja sen palveluita vähentääkseen resurssien käyttöä. (Mavridis ja Karatza, 2021) Virtuaalikoneisiin pohjautuvissa arkkitehtuureissa kapseli voidaan mieltää vastaamaan yhtä virtuaalikonetta, sillä se toteuttaa yleensä yhden kokonaisen sovelluksen. Kapseli voi sisältää useita palveluita, jotka pystyvät viestimään keskenään käyttämällä paikallisen osoiteavaruuden (engl. localhost) osoitteita (Rensin, 2015).

Hallinnan kannalta Kubernetes suorittaa operaatiota aina vähintään kapselitasolla, eli konfiguraatiomuutos yksittäiseen konttiin pakottaa koko kapselin vastaamaan (Rensin, 2015). Käytännössä tämä tarkoittaa siis sitä, että muutoksien tapahtuessa vanhaa konfiguraatiota käyttävät kapselit ajetaan alas kokonaisuudessaan, vaikka vain yksi kontti kaipaisi muutosta. Lisäksi Kubernetes huolehtii aina, että jokainen kapseli on toimintakunnossa ja pystyy itsenäisesti uudelleenkäynnistämään kontteja kapselin sisällä, mikäli niissä havaitaan ongelmia (”Kubernetes Documentation - Pod Lifecycle”, 2022). Tämä takaa, että suunniteltu palvelu toimii aina kokonaisuutena ja pystyy itsenäisesti palautu-

maan virhetilanteista, jos sen suunnitteluvaiheessa on päätetty sijoittaa kaikki sen osapalvelut osaksi yhtä kapselia. Kuberneteksen avulla toteutettuja palveluita voidaan siten myös hallita ja laajentaa helposti luomalla uusia kopioita palvelun toteuttavasta kapselista, sillä itsenäisesti palvelun toteuttava kapseli sisältää kaikki sen tarvittavat osakomponentit.

Palvelu voidaan myös jakaa pienempiin osiin sijoittamalla jokainen kontti omaan kapseliinsa, jolloin niitä voidaan niitä skaalata ja hallita omina osinaan. Toisaalta tämä edellyttää, että kontit kykenevät löytämään toisensa paikallisten osoitteiden ulkopuolella ja että niiden toteuttamat palvelut sietävät väliaikaisia virheitä, mikäli niiden tarvitsemat kontit eivät vielä ole saatavilla.

## 2.2 Jalkautukset

Jalkautukset (engl. deployment) ovat Kuberneteksen tapa esittää järjestelmän toivottu tila konfiguraation muodossa. Jalkautus koostuu YAML-merkintäkielellä (engl. YAML Ain't Markup Language) kirjoitetusta tekstitiedostosta, joka kuvaa järjestelmän halutun tilan tekstinä. Jalkautus voi esimerkiksi kertoa, kuinka monta kapselia tulisi luoda ja mitä kontteja minkäkin kapselin tulisi sisältää. Kun jalkautus ajetaan, Kubernetes etsii sen kuvaamien konttien levykuvat valitusta konttirekisteristä (engl. container registry) ja sijoittaa ne kapseleihin, jotka edelleen sijoitetaan klusterille. (”Kubernetes Documentation - Deployments”, 2022)

Jalkautukset ovat yleisesti suositeltu tapa lisätä kapseleita Kubernetes-ympäristöön. Vaikka yksittäisen kapselin lisääminen on myös mahdollista, ei se ole suositeltua, sillä ilman jalkautusta Kubernetes ei voi hallita kapselin käyttäytymistä. Jalkautuksen ohjain huolehtii, että jalkautuksen kuvaamaa tilaa ylläpidetään. Ongelmatilanteissa ohjain esimerkiksi huolehtii, että vikatilanteeseen päätynyt kapseli korjataan tai korvataan. (”Kubernetes Documentation - Configuration Best Practices”, 2021)

Jalkautuksien avulla Kubernetes voi myös automaattisesti päivittää niiden toteuttaman palvelun ilman käyttökatkoa. Ajamalla jalkautuksen uudelleen päivityskomennolla Kubernetes alkaa luomaan päivitysstrategian mukaisesti uusia kapseleita ja niiden saavuttaessa käyttövalmiuden Kubernetes automaattisesti ajaa alas kapselin vanhan version. (Sayfan, 2020) Päivitysstrategia voidaan kuvata jalkautuksen YAML-tiedostoon. Se määrittää kuinka monta kapselia Kubernetes voi ajaa alas ilman, että uudet kapselit ovat valmiita käyttöön ja kuinka monta uutta kapselia voidaan luoda ennen kuin vanhat kapselit on ajettu alas. (Thelin, 2020)

## 2.3 Tilallisuuden hallinta

Tilallisten sovellusten hallinta Kuberneteksessa on tilattomiin verrattuna hieman hankalampaa. Kubernetes ei alun perin tukenut lainkaan tilallisia sovelluksia ja sen suunnitte-

lufilosofia, jonka ajatuksena on pelkistää palvelimien roolia, sopii parhaiten tilattomille sovelluksille (Sayfan, 2020). Vasta versiossa 1.3 Kubernetes esitteli StatefulSet-toimintoa edeltävän PetSetin alpha-version, mikä mahdollisti tilallisten sovellusten toteuttamisen (”Kubernetes Blog - Kubernetes 1.3: Bridging Cloud Native and Enterprise Workloads”, 2016).

Sovelluksen tilallisuus kuvaa sitä, miten aiemmat operaatiot vaikuttavat tuleviin operaatioihin. Esimerkiksi yksikertaisen verkkosivun tarjoava palvelin lähettää aina saman sivun, kun se saa pyynnön verkosta. Tässä tilanteessa on kyse tilattomasta sovelluksesta, ei ole siis esimerkiksi väliä onko sivun tarjoava palvelin sama kuin sivua aiemmin tarjonnut, lopputulos on aina sama. Tilallisen sovelluksen tapauksessa sivu voisi pitää muistissa jotain tietoa asiakkaastaan, esimerkiksi asiakkaan yhteyden tiedot, jotta se voi lähettää asiakkaalle päivityksiä sisällön muuttuessa. Jos sovellus käynnistyy uudelleen, se menettää tämän yhteyden tiedot, ellei niitä erikseen tallenneta. (N. D. Nguyen ja Kim, 2021)

Tilallisia sovelluksia suunnitellessa pitäisi pitää mielessään, että Kubernetes ei pidä kapseleita tilallisina. Kubernetes saattaa normaalikonfiguraatiossaan, tilanteen sitä vaatiessa, siirtää kapseleita noodilta (engl. node), eli usein fyysiseltä palvelimelta, toiselle tuhoamalla alkuperäisen ja luomalla uuden kapselin sille annetun konfiguraation mukaan. (Rensin, 2015) Tällöin kaikki välimuistissa oleva tilallinen informaatio alkuperäisestä kapselista menetetään. Koska oletusarvoisesti Kubernetes ei luo pysyviä levyjä konteille, myös sovellusten levylle kirjaama tieto menetetään kapselin tuhoutuessa.

Jotta Kuberneteksen avulla voitaisiin hallita tilallisia sovelluksia, tulisi sen tarjota työkalut, joiden avulla voidaan taata, että sovelluksen tila säästyy uudelleenluomisen tapahtuessa. Käytännössä tämä siis tarkoittaa, että Kuberneteksen tulisi kyetä tallentamaan tietoa pysyvästi levylle, sekä ratkaista jollakin tavalla menetettävän välimuistin ongelma.

### 2.3.1 StatefulSet ja taltiot

StatefulSet on jalkautuksen kaltainen työyksikkö, joka jalkautuksen tapaan esittää järjestelmän toivotun tilan YAML-tekstitiedoston avulla. Jalkautuksesta poiketen StatefulSet on tarkoitettu tilallisia sovelluksia varten ja takaa sen luomille kapseleille uniikin tunnuksen ja pysyvän levytilan. Kapselin uniikki tunnuksesta ja levytilasta muodostuva identiteetti pysyy aina kapselin mukana riippumatta mille noodille se sijoitetaan. (”Kubernetes Documentation - StatefulSets”, 2022)

Levytila, jonka StatefulSet kapseleille tarjoaa, luodaan joko manuaalisesti Kuberneteksen käyttöliittymän kautta tai automaattisesti PersistentVolume Provisioner-komponentin avulla. Varattua määrällistä levytilaa kutsutaan taltioksi (engl. volume). Kubernetes ei automaattisesti koskaan tuhoa luotuja taltioita, joten niiden sisältämä tilallinen informaatio säilyy. (”Kubernetes Documentation - StatefulSets”, 2022) StatefulSettien ja niiden luo-

mien taltioiden käyttö takaa siis sovellukselle uniikin identiteetin ja siihen sidotun tilallisen informaation. Uniikki identiteetti voidaan sitoa verkkotunnukseen niin, että sama tunnus vastaa aina samaa kapselia. Jos sovelluksen tila onnistutaan myös kuvaamaan levylle missä tahansa tilanteessa, voidaan katsoa tilallisuuden säilyttämisen onnistuneen. Tallentaminen on kuitenkin vain erityisen tärkeää juuri sillä hetkellä, kun kapseli tuhotaan ja luodaan uudelleen. Tätä varten tarvitaan avuksi kapselin elinkaaren hallinnan tarjoamia työkaluja.

### 2.3.2 Elinkaaren hallinta

Kapselin elinkaaren lopussa eli kapselin tuhoamisen yhteydessä Kubernetes lähettää kapselin konttien pääprosessille TERM-signaalin. TERM-signaalin antamisella on tarkoitus välttää ohjelmien äkillinen keskeytys ja antaa niille aikaa reagoida sammuttamistarpeeseen. (”Kubernetes Documentation - Pods”, 2021) Useimmille ohjelmille tämä on riittävä ratkaisu välimuistin menettämisen ratkaisuun. Mikäli ohjelma osaa reagoida TERM-signaaliin tallentamalla kaiken välimuistissa olevan tiedon levylle, se voidaan myöhemmin käynnistää toisessa kapselissa menettämättä informaatiota.

Kaikki ohjelmat eivät kuitenkaan välttämättä reagoi TERM-signaaliin toivotulla tavalla. Ohjelma voi esimerkiksi sulkea avoimet verkkoyhteydet toivotusti, mutta se ei tallenna tilaansa levylle ilman erillistä komentoa. Tätä varten Kubernetes tarjoaa konfiguroitavan preStop-koukun (engl. preStop hook), joka suoritetaan ennen TERM-signaalin lähettämistä. PreStop-koukun avulla kapselin kontit voidaan määrittää ajamaan mielivaltaisia komentoja tai skriptejä, joiden avulla tila voidaan tallentaa. (”Kubernetes Documentation - Pods”, 2021)

# 3. KLUSTERIT

Klusterit koostuvat yhdestä tai useammasta noodista, jotka voivat olla fyysisiä tai virtuaalikoneita. Kun jalkautus ajetaan, Kubernetes toteuttaa sen kuvaamat resurssit jollekin klusterin noodeista. (”Kubernetes Documentation - Nodes”, 2022) Kubernetes valitsee mille noodille resurssit sijoitetaan ensisijaisesti konfiguraation määritelmien mukaan, mutta määritelmien puuttuessa hyödyntää sisäisiä algoritmeja optimaalisen noodin löytämiseen. Yleisimmissä tilanteissa Kubernetes hallitsee kerrallaan yhden klusterin toimintaa.

Kuberneteksen käytön yleistyessä yhä useammin yksi klusteri ei kuitenkaan enää riitä, vaan klustereita halutaan toteuttaa useille palveluntarjoajille ja maantieteellisille alueille. Syitä tähän voi olla monia. Palveluntarjoajat voivat esimerkiksi tarjota palvelimia, joiden suorituskyky on toisistaan poikkeava. Esimerkiksi verrattain suosittu palveluntarjoaja DigitalOcean hinnoittelee korkeamman suorituskyvyn tarjoavat virtuaalikoneet dedikoiduilla prosessoriytimillä huomattavasti yleiskäyttöön tarkoitettuja virtuaalikoneita korkeammalle (”DigitalOcean - Droplet Pricing”, 2022). Tarve sijoittaa klusterit eri maantieteellisille alueille voi johtaa syynsä esimerkiksi paremmasta vasteajasta kuluttajille tai sitä voidaan edellyttää lainsäädännön vuoksi. Esimerkiksi Euroopan unionin GDPR-lainsäädäntö edellyttää, että EU-kansalaisesta kerätyt tiedot säilytetään fyysisesti EU-alueella. (Reselman, 2021)

Kubernetes ei ilman lisäosia tue yhdistettyjä klustereita. Useamman klusterin järjestelmiä voidaan silti toteuttaa ja eri ohjelmistoja hyödyntämällä myös yhdistetyt klusterit ovat mahdollisia. (Arbezzano ja Palesandro, 2021)

## 3.1 Erilliset useamman klusterin järjestelmät

Yksinkertaisimmillaan useamman klusterin järjestelmä voidaan toteuttaa manuaalisesti täysin erillisillä klustereilla. Tällöin käytössä on kaksi tai useampi erillistä klusteria, jotka kaikki toteuttavat Kuberneteksen koko ohjelmistopinon toisistaan riippumatta. Klustereiden erillisyys tarjoaa joitakin hyötyjä verrattaessa yhteen tai yhdistettyihin klustereihin. Esimerkiksi mikäli virheellinen konfiguraatio aiheuttaa yhdellä klusterilla ongelman, toinen klusteri on vielä toimintakunnossa. Tämä tietenkin edellyttää, ettei klustereille samanaikaisesti ajeta virheitä aiheuttavia komentoja tai että niiden tila ei ole niin samankaltainen, että ne päätyisivät samaan virhetilaan.

Erilliset useamman klusterin järjestelmät voivat myös helpottaa ohjelmistojen sijoituksen hallintaa heterogeenisissa klustereissa. Heterogeeniset klusterit koostuvat noodeista, jotka eivät ole identtisiä. Ne voivat esimerkiksi sisältää noodeja, joiden prosessoriarkkitehtuuri tai suorituskyky on muista noodeista poikkeava. (El Haj Ahmed et al., 2021) Kubernetes tarjoaa tätä varten nodeSelector-rajauksen, jonka avulla kapselit voidaan määrittää ajettavaksi vain tietyille noodeille (”Kubernetes Documentation - Assigning Pods to Nodes”, 2022). NodeSelector-rajauksen käyttäminen vaikeuttaa klusterin käyttöä, sillä jokaiselle kapselille pitää määrittää mitä noodeja se suosii, jos tätä rajausta tarvitaan. Joissakin tilanteissa voi olla yksinkertaisempaa vain jakaa erilaiset noodit erillisiin klustereihin, jotta konfiguraatioiden monimutkaistamiselta vältytään.

Erillisten klustereiden järjestelmien suurin ongelma on niiden hallinta, ilman keskitettyä hallintaratkaisua hallintatyö moninkertaistuu klustereiden määrän mukaan. Jotta tältä vältyttäisiin, on hallintaa varten luotu useita eri työkaluja, jotka yleensä pyrkivät keskittämään hallintatehtävät yhteen käyttöliittymään.

## 3.2 Keskitetyt useamman klusterin järjestelmät

Keskitetty hallinta pyrkii ratkaisemaan täysin toisistaan erillään olevien klustereiden suurimmat ongelmat pitäen silti niiden hyötyjä. Keskitetyn hallinnan pääidea on nimensä mukaisesti keskittää eri klustereiden hallinta yhteen käyttöliittymään. Klusterit voidaan edelleen pitää erillään toisistaan, mutta jos niihin kohdistuvat hallintatehtävät ovat samankaltaisia, voidaan niitä esimerkiksi ajaa useammalle klusterilla samaan aikaan. Kubernetes tarjoaa minimaalisen tuen keskitettyyn hallintaan, sillä se tukee useampaan klusteriin yhdistämistä kubectl-komentorivityökalun kautta (”Kubernetes Documentation - Organizing Cluster Access Using kubeconfig Files”, 2022).

Kuberneteksen laajennettavuuden ja avoimen rajapinnan vuoksi on luotu useita ohjelmistoja, jotka parantavat jo olemassa olevia toimintoja. Koska keskitetty hallinta on muodostunut tärkeäksi osa-alueeksi Kubernetes-ympäristöjen kasvaessa, sitä varten on luonnollisesti luotu useita eri ohjelmistoja. (Arbezzano ja Palesandro, 2021)

Suosittu Rancher-ohjelmisto (”Rancher”, 2022) on eräs vaihtoehto useamman klusterin keskitettyyn hallintaan. Se tarjoaa kubectl-työkalusta poiketen graafisen käyttöliittymän, sekä useita hyödyllisiä toimintoja, kuten keskitetyn lokitiedostojen valvonnan ja sovelluskatalogin. Keskitetyn hallinnan työkalut voivat myös hyödyntää ulkoisten palveluiden rajapintoja keskittääkseen hallintatehtäviä entisestään. Esimerkiksi Rancher pystyy kutsumaan useiden palveluntarjoajien, kuten Amazonin, Googlen ja Microsoftin pilvipohjaisten Kubernetes-palveluiden rajapintatoimintoja ja luomaan siten uusia klustereita ilman, että käyttäjän tarvitsisi tehdä niin manuaalisesti kunkin palveluntarjoajan omassa verkkokäyttöliittymässä (”Rancher Documentation - Setting up Clusters from Hosted Kubernetes Providers”, 2021).

## 3.3 Federoidut useamman klusterin järjestelmät

KubeFed on eräs hallintajärjestelmä usean Kubernetes-klusterin järjestelmille. Se keskittää useamman klusterin hallinnan yhdelle pääklusterille ja mahdollistaa useamman klusterin hallinnan yhden rajapinnan kautta. KubeFed ottaa vastaan komentoja pääklusterin kautta ja välittää sen kautta muille osaklustereille kaikki niitä koskevat konfiguraatiomuutokset. (”KubeFed”, 2021) KubeFedin avulla yhdistettyjä klustereita kutsutaan federoiduiksi klustereiksi, sillä niitä hallitsee yksi hallitsijaklusteri, mutta kaikki klusterit vastaavat silti itsenäisesti monista toiminnoistaan.

KubeFedin tarkoitus on luoda federoituja versioita olemassa olevista rajapintatoiminnoista, eli siis luoda uusi rajapintatoiminto, jota KubeFed hallitsee ja joka toteuttaa Kuberneteksen rajapinnan toiminnon usealle klusterille jaettuna. Esimerkiksi KubeFedin avulla voidaan luoda federoitu jalkautustoiminto, eli FederatedDeployment. Kun federoitu jalkautus ajetaan, oletusarvoisesti se sijoittaa jalkautuksen kapselit tasaisesti jokaiselle klusterille. Esimerkiksi kahden klusterin federaatiossa kuuden kapselin federoitu jalkautus toteuttaa kolmen kapselin jalkautuksen molemmille klustereille. (”KubeFed user guide”, 2021)

KubeFedin avulla moni keskeinen Kuberneteksen rajapintatoiminto voidaan muokata federoituun muotoon. Konfiguraatiossa voidaan myös käyttää tavallisia rajapintatoimintoja samaan aikaan federoitujen kanssa. Täten lähes mistä tahansa alkuperäisestä yhden klusterin konfiguraatiosta voidaan toteuttaa versio, joka toimii federoidussa ympäristössä. (”KubeFed user guide”, 2021) KubeFed toimii palveluntarjoajasta riippumattomana, mikä tarkoittaa, että se kutsuu ainoastaan Kuberneteksen toteuttamia rajapintatoimintoja, eikä ole tietoinen palvelutarjoajien omista laajennuksista, mikäli sellaisia on toteutettu. Tämä tarkoittaa, että KubeFed ei lukitse käyttäjää tiettyyn palveluntarjoajaan, mutta toisaalta osa hyödyllisistä toiminnoista, kuten uusien klusterien luominen pilvipohjaiseen palveluun yksinomaan KubeFedia käyttäen ei ole mahdollista.

# 4. SKAALAAMINEN

Kuberneteksen suurimpia hyötyjä hallintatehtävissä on sen kyky vastata automaattisesti muuttuviin resurssitarpeisiin. Kubernetes voi tarvittaessa skaalata kapseleiden määrää ja useimmissa pilvipohjaisissa Kubernetes-ympäristöissä myös noodeja voidaan lisätä automaattisesti.

## 4.1 Horizontal Pod Autoscaler

Yleisin skaalaamistarve syntyy, kun hallittavan järjestelmän tulisi vastata nousevaan käyttöasteeseen. Perinteiset staattisen kapasiteetin järjestelmät eivät skaalaannu, vaan käyttöasteen noustessa niiden kuormitus nousee ja lopulta ylittyy, jolloin palvelun laatu heikkenee. Manuaalinen skaalaaminen, eli esimerkiksi uusien virtuaalipalvelimien luominen käsityönä, on hidasta ja vastaa kuormaan viiveellä. Tarvittaisiin siis tapa, jolla järjestelmiä voitaisiin skaalata automaattisesti, kun niiden käyttöaste nousee yli jonkin asetetun rajan.

Kuberneteksen Horizontal Pod Autoscaler (HPA) vastaa kapseleiden vaakasuuntaisesta skaalaamisesta eli se lisää kopioita kapseleista järjestelmään, kun tarve niille syntyy. HPA määrittelee tarpeen lisäkapseleille pääasiassa niiden resurssien käyttöasteen mukaan, joskin rajapinta mahdollistaa vapaasti mukautettujen metriikoiden käytön. (”Kubernetes Documentation - Horizontal Pod Autoscaling”, 2022) Yleisiä palvelun käyttöasteesta kertovia metriikoita ovat esimerkiksi muistin ja prosessorin käyttöaste. Voidaan esimerkiksi määrittää, että kapselin halutaan käyttävän alle 200 megatavua muistia. Kun tämä raja ylittyy, Kubernetes pyrkii kasvattamaan järjestelmän kapasiteettia luomalla lisää kapseleita, kunnes muistin keskimääräinen käyttöaste vastaa määritelmää (T.-T. Nguyen et al., 2020).

Federoiduissa klustereissa HPA voidaan määrittää federoiduksi tai sitä voidaan käyttää ilman federointia yksittäisissä klustereissa. Kuten yhden klusterin tapauksessa, federoitu HPA huolehtii, että kapseleita on käyttöasteeseen nähden riittävä määrä. Federoitu HPA kuitenkin voi sijoittaa kapseleita kaikille federoidun klusterin osaklustereille, kun taas tavallinen HPA on rajattu omaan klusteriinsa. (”KubeFed user guide”, 2021)

Federoidun Horizontal Pod Autoscalerin tapaa sijoittaa uusia kapseleita eri klustereille voidaan ohjata KubeFedin Replica Scheduling Preferencen (RSP) avulla. Tyhjä konfigu-

raatio määrittää kapselit jaettavaksi tasan kaikkien federaation osaklustereiden kesken, mutta hyödyllisempää on määrittää RSP:lle painoarvot. Antamalla eri klustereille erisuuruiset painoarvot voidaan määrittää KubeFed suosimaan tiettyjä klustereita. Klustereille voidaan myös määrittää minimi- ja maksimiarvot kapselikopioiden määrälle. (”KubeFed user guide”, 2021)

## 4.2 Cluster Autoscaler

Kubernetes sijoittaa kapseleita noodeille niin, että niille konfiguroidut resurssipyynnöt saadaan toteutettua (Medel et al., 2017). On kuitenkin huomioitava, että resurssipyyntö eroaa resurssirajoitteesta ja että Kubernetes ei huomioi resurssirajoitteita sijoittamisen yhteydessä (”Kubernetes Documentation - Resource Management for Pods and Containers”, 2022). Mikäli noodille sijoitettujen kapseleiden resurssien käyttö ylittää niille esitetyt resurssipyynnöt, mutta resurssirajat eivät ylity, noodi päätyy ylisitoutuneeseen (engl. overcommitted) tilaan, jossa suorituskyky kärsii resurssien puutteen takia. Esimerkiksi muistin suhteen noodi voi joutua sijoittamaan osan ylikäytetystä muistista kovalevyllä sijaitsevaan swap-muistiin, jonka suorituskyky on oikeaa muistia selkeästi huonompi (Ro, 2014).

Noodien ylikuormitus on ongelma, jolle tarvitaan tehokas ratkaisu. Ilman lisänoodeja täysi järjestelmä ei kykene enää skaalaamaan kapseleita, mutta jos noodien määrä on mitoitettu aina suurimman tarpeen mukaan, järjestelmä varaa turhaan ylimääräisiä resursseja. Monet pilvipohjaiset Kubernetes-ympäristöt, kuten Googlen GKE (”Google Kubernetes Engine”, 2022) ja Amazonin EKS (”Amazon Elastic Kubernetes Service”, 2022) tarjoavat ratkaisuksi skaalattavan klusterin. Pilvipalveluissa Kuberneteksen noodit ovat yleensä toteutettu virtuaalikoneina, jolloin niiden luominen ja klusteriin lisääminen on verrattain nopeaa. Cluster Autoscaler-komponentit huolehtivat, että klusterilla on aina riittävästi resursseja kapseleiden luomiseen luomalla uusia noodeja resurssien käyttöasteen kasvaessa (Poniszewska-Marańda ja Czechowska, 2021).

Kubernetes tarjoaa myös oman toteutuksensa automaattisesta skaalaimesta klustereille. Kuberneteksen oma skaalain tarkistaa skaalaustarpeen tasaisin aikavälein ja pilvipohjaisten skaalainten tapaan resurssien puutetilan ilmentyessä se laajentaa klusteria. Skaalaimesta on toteutettu versiot lähes kaikille suuremmille pilvipalveluiden tarjoajille, joten se kykenee skaalaamaan klustereita lähes missä tahansa yleisessä pilvipohjaisessa Kubernetes-toteutuksessa. (”Kubernetes Autoscaler”, 2022)

Klusterin automaattinen skaalaaminen toimii KubeFedistä erillään ja KubeFed ei ota kantaa klusterin koon muutoksiin. Automaattinen klusterin skaalaaminen on luonnollinen lisä ympäristöön, joka hyötyy KubeFedin tarjoamasta automaatiosta kapseleiden sijoittamisen suhteen. Automaattinen skaalaus mahdollistaa KubeFedin kanssa ympäristön, jossa oletettua pysyvää kuormitusta varten voidaan toteuttaa kustannustehokas paikallinen

(engl. On-Premises) palvelinratkaisu ja muuttuvan kuormituksen osio voidaan kattaa automaattisesti skaalautuvalla pilvipohjaisella Kubernetes-ympäristöllä (Fisher, 2018). Tässä ympäristössä KubeFed mahdollistaa keskitetyn hallinnan, joka automaattisesti skaalaa pilvipohjaisen kapasiteetin alaspäin kuormituksen laskiessa ja pystyy myös sen jälkeen skaalaamaan paikallisen ympäristön kapseleiden määrää. Tilanteessa, jossa paikallisen ympäristön kuormitus laskee odotettua tasoa alemmaksi, voi olla hyödyllistä skaalata järjestelmän kapselikopioita alaspäin esimerkiksi energian säästämiseksi.

# 5. YHTEENVETO

Tässä tutkielmassa tutkittiin, miten tilallisia sovelluksia voidaan skaalata federoiduissa Kubernetes-klustereissa. Kubernetes toimii parhaimmin tilattomien sovellusten hallinnassa, mutta se tarjoaa myös työkalut, joiden avulla tilallisia sovelluksia pystytään tehokkaasti hallitsemaan. StatefulSet mahdollistaa kapseleiden yksilöinnin ja pysyvät taltiot mahdollistavat tiedon kestävän tallentamisen. Kapselin elinkaaren hallinnan avulla voidaan huolehtia siitä, että kapseli ehtii tallentaa tilansa ennen sen tuhoamista tai siirtämistä.

Kubernetes-federaatio on Kuberneteksen klusterityyppi, jossa useampaa klusteria voidaan hallita yhden pääklusterin kautta. Federaatioklusterit ovat yksittäiseen klusteriin nähden vikasietoisempia, sillä ne voivat hyödyntää useampia palveluntarjoajia samanaikaisesti. Kubernetes-federaation toteuttaa KubeFed-ohjelmisto, joka voidaan erikseen asentaa tavalliseen Kubernetes-klusteriin.

KubeFed mahdollistaa useimpien Kuberneteksen rajapinnan tarjoamien toimintojen lisäämisen federaation hallittavaksi. Pääklusteri voi näiden federoitujen toimintojen kautta hallita kapseleiden luomista jokaisella federaatioon kuuluvalla klusterilla. Sovellusten skaalaamisen voi toteuttaa joko pääklusterin kautta tai antaa federaation klustereiden huolehtia sovelluksen skaalaamisesta itse.

Tutkielman aineistoa valittaessa huomattiin, että usein ensisijainen lähde saatiin Kuberneteksen omasta dokumentaatiosta. Dokumentaatio hyvin harvoin esittää väitteiden takana olevaa teoriaa tai antaa lähteitä, joten teorian eristäminen lähempänä käytäntöä olevista ohjeista oli usein hankalaa. Monessa akateemisessa julkaisussa esitetyt tutkimuskohteet eivät olleet tämän tutkielman kannalta relevantteja, joten luotettavien vertaisarvioiden lähteiden löytäminen oli työlästä ja usein viitattu osuus oli vain pieni osa koko julkaisun sisällöstä. Kuberneteksessa ja federoiduissa klustereissa monet suunnittelupäätökset varmasti perustuvat johonkin tieteellisesti varmistettavaan havaintoon, joten näiden lähteiden löytyminen dokumentaatiosta olisi ollut hyödyllistä.

Tämä tutkielma keskittyi tarkemmin vain federoituun Kubernetes-ympäristöön, mutta tutkielman edetessä huomattiin, että klustereita voidaan toteuttaa useiden eri ohjelmistojen avulla, joissa hallinnan ongelmaa lähestytään useista eri näkökulmista. Tämän tutkielman pohjatietoja voisi siis myös myöhemmin hyödyntää muiden ratkaisujen tarkastelussa tai kattavamman vertailun toteuttamisessa.

# LÄHTEET

*Amazon elastic kubernetes service*. (2022). Retrieved May 19, 2022, from https://aws.amazon.com/eks/

Arbezzano, G., & Palesandro, A. (2021). *Cncf blog - simplifying multi-clusters in kubernetes*. Retrieved June 7, 2022, from https://www.cncf.io/blog/2021/04/12/simplifying-multi-clusters-in-kubernetes/

*Digitalocean - droplet pricing*. (2022). Retrieved May 29, 2022, from https://www.digitalocean.com/pricing/droplets

El Haj Ahmed, G., Gil-Castiñeira, F., & Costa-Montenegro, E. (2021). Kubcg: A dynamic kubernetes scheduler for heterogeneous clusters. *Software, practice I& experience*, *51*(2), 213–234.

Fisher, C. (2018). Cloud versus on-premise computing. *American Journal of Industrial and Business Management*, *8*(9), 1991–2006.

*Google kubernetes engine*. (2022). Retrieved May 19, 2022, from https://cloud.google.com/kubernetes-engine

*Kubefed*. (2021). Retrieved May 27, 2022, from https://github.com/kubernetes-sigs/kubefed

*Kubefed user guide*. (2021). Retrieved May 29, 2022, from https://github.com/kubernetes-sigs/kubefed/blob/master/docs/userguide.md

*Kubernetes*. (2022). Retrieved March 10, 2022, from https://kubernetes.io/

*Kubernetes autoscaler*. (2022). Retrieved August 30, 2022, from https://github.com/kubernetes/autoscaler

*Kubernetes blog - kubernetes 1.3: Bridging cloud native and enterprise workloads*. (2016). Retrieved June 6, 2022, from https://kubernetes.io/blog/2016/07/kubernetes-1-3-bridging-cloud-native-and-enterprise-workloads/

*Kubernetes documentation - assigning pods to nodes*. (2022). https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/

*Kubernetes documentation - configuration best practices*. (2021). Retrieved June 8, 2022, from https://kubernetes.io/docs/concepts/configuration/overview/

*Kubernetes documentation - deployments*. (2022). Retrieved March 19, 2022, from https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

*Kubernetes documentation - horizontal pod autoscaling*. (2022). Retrieved May 10, 2022, from https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

*Kubernetes documentation - nodes*. (2022). Retrieved March 19, 2022, from https://kubernetes.io/docs/concepts/architecture/nodes/

*Kubernetes documentation - organizing cluster access using kubeconfig files*. (2022). Retrieved August 30, 2022, from https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/

*Kubernetes documentation - pod lifecycle*. (2022). Retrieved August 31, 2022, from https://kubernetes.io/docs/concepts/architecture/nodes/

*Kubernetes documentation - pods*. (2021). Retrieved February 20, 2022, from https://kubernetes.io/docs/concepts/workloads/pods/

*Kubernetes documentation - resource management for pods and containers*. (2022). Retrieved May 17, 2022, from https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

*Kubernetes documentation - statefulsets*. (2022). Retrieved May 21, 2022, from https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Mavridis, I., & Karatza, H. (2021). Orchestrated sandboxed containers, unikernels, and virtual machines for isolation-enhanced multitenant workloads and serverless computing in cloud. *Concurrency and computation*.

Medel, V., Tolón, C., Arronategui, U., Tolosana-Calasanz, R., Bañares, J. Á., & Rana, O. F. (2017). Client-side scheduling based on application characterization on kubernetes. *Economics of grids, clouds, systems, and services* (pp. 162–176). Springer International Publishing.

Nguyen, N. D., & Kim, T. (2021). Balanced leader distribution algorithm in kubernetes clusters. *Sensors (Basel, Switzerland)*, *21*(3), 1–15.

Nguyen, T.-T., Yeom, Y.-J., Kim, T., Park, D.-H., & Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors (Basel, Switzerland)*, *20*(16), 1–18.

Poniszewska-Marańda, A., & Czechowska, E. (2021). Kubernetes cluster for automating software production environment. *Sensors (Basel, Switzerland)*, *21*(5), 1–24.

*Rancher*. (2022). Retrieved August 25, 2022, from https://rancher.com/

*Rancher documentation - setting up clusters from hosted kubernetes providers*. (2021). Retrieved August 30, 2022, from https://rancher.com/docs/rancher/v2.6/en/cluster-provisioning/hosted-kubernetes-clusters/

Rensin, D. K. (2015). *Kubernetes*. O'Reilly Media, Inc.

Reselman, B. (2021). *Multi-cluster kubernetes architecture*. Retrieved August 28, 2022, from https://www.redhat.com/architect/multi-cluster-kubernetes-architecture

Ro, C. (2014). Modeling and analysis of memory virtualization in cloud computing. *Cluster computing*, *18*(1), 177–185.

Sayfan, G. (2020). *Mastering kubernetes - third edition* (3rd edition). Packt Publishing.

*Techcrunch - as kubernetes hits 1.0, google donates technology to newly formed cloud native computing foundation*. (2015). Retrieved March 5, 2022, from https://techcrunch.com/2015/07/21/as-kubernetes-hits-1-0-google-donates-technology-

to-newly-formed-cloud-native-computing-foundation-with-ibm-intel-twitter-and-others/

Thelin, R. (2020). *A deep dive into kubernetes deployment strategies*. Retrieved July 24, 2022, from https://www.educative.io/blog/kubernetes-deployments-strategies

Verma, A., Pedrosa, L., Korupolu, D., Madhukar Oppenheimer, Tune, E., & Wilkes, J. Large-scale cluster management at google with borg. In: Proceedings of the 10th European Conference on Computer Systems, EuroSys 2015, 2015-04-17, 2015.